A3: DIVIDE AND CONQUER, MASTER METHOD, MORE RECURRENCES.

"Divide and rule, weaken and conquer, love and enslave, these are three tenets of politics" — Bangambiki Habyarimana

Course: CS 5006
Summer 2018
Due: 25 May 2018, 5pm

OBJECTIVES

After you complete this assignment, you will be comfortable with:

- Using the Master Method for solving recurrences
- Choosing methods for solving recurrences
- Applying Divide and Conquer
- Recursion details
- Using recurrences to calculate worst-case analysis of a given algorithm.

RELEVANT READING

- Kleinberg and Tardos, Chapter 5

AND NOW ON TO THE FUN STUFF.

Problem 1: Stable Sorting (4 points)

Stable sorting algorithms leave equal key items in the same relative order as in the original permutation. Explain what must be done to ensure that mergesort is a stable sorting algorithm.

You probably want to have a part in the mergesort algorithm that says, after you've split the array into it's original components, if A[i] = B[i], with A being the portion on the left, and B being the portion on the right:

ArrayYouAreRearrangingThemInto[Current Free Location in Array] = A[i];
Key = A[i+1];
Mergesort while portion (Key, B).

So A[i] goes in first, and then you continue comparing and doing mergesort from A with all of B but from A[i+1] onwards.

Problem 2: 20 Qestions (4 points)

Consider the numerical 20 Q!Jestions game. In this game, Player 1 thinks of a number in the range 1 to n. Player 2 has to figure out this number by asking the fewest number of true/false questions. Assume that nobody cheats.

(a) (2 points) What is an optimal strategy if n is known?

If n is known, do binary search. So divide n by 2, then ask whether that value is the value. Based on whether it's higher or lower than that mid value, you will go to that portion of the original array on. So if it's lower, you are continuing binary search from 0 to that mid point you found, and if it's higher, you are doing binary search from the midpoint + 1 to n. Keep doing this until you find the number.

(b) (2 points) What is a good strategy if n is not known?

If n isn't know, ask what you'd think would be a good upper limit as your first number. If the other player says lower, halve that number and do binary search as described in answer A. If, however, the number is higher, double the value of the initial number you asked and then find the midpoint between the initial number you asked about +1, and double the initial amount. Ask about that midpoint. Continue as described above until you receive a response of lower. Then from that point on, do a normal binary search with the starting lower limit being the number you suggested before this most recent one.

Problem 3: Multiplying Polynomials (3 points)

Show how to multiply two linear polynomials $ax + b$ and $cx + d$ using only three multiplications. (Hint: One of the multiplications is $(a + b) \cdot (c + d)$)

Problem 4: Recurrences (14 points)

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

(a) (2 points) $T(n) = 2T(n/2) + n^4$

$T(n) = 2T\left(\frac{n}{2}\right) + n^4; f(n) = n^4; a = 2; b = 2; n^{\log_2 2} = n^1 = n; n^4 > n, \mathit{so\ this\ is\ case}\ 3\ \mathit{of\ master\ method}; T(n) = \Theta(n^4);$

So with this tight bound, upper bound would be $O(n^5)$; and lower bound $\Omega(n^3)$;

(b) (2 points) $T(n) = T(7n/10) + n$

$T(n) = T\left(\frac{7n}{10}\right) + n; f(n) = n; a = 1; b = 10/7; n^{\log_{10/7} 1} = n^0 = 1; n > 1, \mathit{so\ this\ is\ case}\ 3\ \mathit{of\ master\ method};$

$T(n) = \Theta(n);$

regularity condition; $af\left(\frac{n}{b}\right) \le c * f(n)\ \mathit{with}\ c < 1; 1 * \frac{7n}{10} \le c * n; \frac{7n}{10} \le cn;$

$\mathit{if}\ c = \frac{7}{10}\ \mathit{which\ is\ less\ than}\ 1, \mathit{then\ this\ passes\ the\ regulatory\ condition}.$ Big O $= O(n^2)$, Big Ω $= \Omega(\log n)$

(c) (2 points) $T(n) = 16T(n/4) + n^2$

$T(n) = 16T\left(\frac{n}{4}\right) + n^2; f(n) = n^2; a = 16; b = 4; n^{\log_4 16} = n^2; f(n) = n^2 = n^{\log_4 16};$

$\mathit{this\ is\ an\ example\ of\ case}\ 2\ \mathit{so}, T(n) = \Theta(n^{\log_4 16} \lg n) = \Theta(n^2 \lg n);$

$\mathit{so\ upper\ bound\ would\ be}\ O(n^3)\ \mathit{and\ lower\ bound\ would\ be}\ \Omega(n^2)$

(d) (2 points) $T(n) = 7T(n/3) + n^2$

$T(n) = 7T\left(\frac{n}{3}\right) + n^2; f(n) = n^2; a = 7; b = 3; n^{\log_3 7} = n^{\frac{\log 7}{\log 3}} = n^{1.77\cdots}; f(n) = n^2 > n^{1.77\cdots};$

$\mathit{this\ is\ an\ example\ of\ case}\ 3\ \mathit{so}, T(n) = \Theta(f(n)) = \Theta(n^2);$

$\mathit{so\ upper\ bound\ would\ be}\ O(n^3)\ \mathit{and\ lower\ bound\ would\ be}\ \Omega(n^2)$

(e) (2 points) $T(n) = 7T(n/2) + n^2$

$T(n) = 7T\left(\frac{n}{2}\right) + n^2; f(n) = n^2; a = 7; b = 2;$

$n^{\log_2 7} = n^{\frac{\log 7}{\log 2}} = n^{2.80\cdots}..; n^2 < n^{2.80\cdots}; \mathit{this\ is\ case}\ 1; \mathit{so}\ T(n) = \Theta\left(n^{\log_2 7}\right) = \Theta(n^{2.80\cdots})$

Big O = $O(n^3)$; Big Ω = $\Omega(n^2)$

(f) (2 points) $T(n) = 2T(n/4) + \sqrt{n}$

$T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}; f(n) = \sqrt{n}; a = 2; b = 4; n^{\log_4 2} = n^{\frac{1}{2}} = \sqrt{n} = f(n);$

$\mathit{this\ is\ case}\ 2, T(n) = \Theta\left(n^{\log_4 2} \lg n\right) = \Theta(\sqrt{n} \lg n)$

(g) (2 points) $T(n) = T(n/2) + n^2$

$$T(n) = T\left(\frac{n}{2}\right) + n^2; f(n) = n^2; a = 1; b = 2; n^{\log_2 1} = n^0 = 1; n^2 > 1;$$

$this\ is\ case\ 3, check\ regularity\ condition{:}\ 1 * f\left(\frac{n}{2}\right) \le c * f(n); \left(\frac{n}{2}\right)^2 \le c * n^2; if\ c = \frac{1}{2}, \frac{n^2}{4} \le \frac{n^2}{2}; so\ it\ fits\ the\ condition.$ $T(n) = \Theta(f(n)) = \Theta(n^2)$. Upper bound - $O(n^3)$. Lower bound - $\Theta(n)$

Problem 5: Chip testing (9 points)

Professor Diogenes has n supposedly identical integrated-circuit (IC) chips that in principle are capable of testing each other. The test jig accommodates two chips at a time. When the jig is loaded, each chip tests the other and reports whether it is good or bad. A good chip always reports accurately whether the other chip is good or bad, but the professor cannot trust the answer of a bad chip. Thus, the four possible outcomes of a test are as follows:

Chip A says	Chip B says	Conclusion
B is good	A is good	both are good, or both are bad
B is good	A is bad	at least one is bad
B is bad	A is good	at least one is bad
B is bad	A is bad	at least one is bad

(a) (3 points) Show that if more than $n/2$ chips are bad, the professor cannot necessarily determine which chips are good using any strategy based on this kind of pairwise test. Assume that the bad chips can conspire to fool the professor.

When n/2 chips could be bad, for the first slot, you have an 50% chance of either good or bad ending up there and for the second lot, depending on whether good or bad went in, there are n/2 chips that could go in for the chip type that wasn't picked and $\frac{n}{2} - 1$ chances for the type that wasn't pick, which is still only 1 less chip than the other type so you're still close to a 50% chance of either happening. As you go past n/2 chips being bad, the 50% chance gets smaller, So there's a greater chance of getting bad chips, so there is a smaller chance the first or second one will be good, so you can't determine whether the chip is good based on the other Chip.

(b) (3 points) Consider the problem of finding a single good chip from among n chips, assuming that more than $n/2$ of the chips are good. Show that $\lfloor n/2 \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size.

If more than n/2 chips are good, there is a greater chance that at least one of the chips is good. So you can decrease the amount of chips by 2 and the as long as the ratio of good chips to bad ones don't drop to below 50%, you should still be able to use the pairwise tests sufficients.

(c) (3 points) Show that the good chips can be identified with $\Theta(n)$ pairwise tests, assuming that more than $n/2$ of the chips are good. Give and solve the recurrence that describes the number of tests.

$$T(n) = T\left(> \frac{n}{2}\right) + T\left(< \frac{n}{2}\right);$$
$$Guess{:}\ \Theta(n);$$
$$T(n) \le c * n; T\left(> \frac{n}{2}\right) \le c\ \left(> \frac{n}{2}\right) = \frac{cn}{2}; if\ c \ge 2, T(n) = n$$

Q!Jestion	Points	Score
Stable Sorting	4	
20 Q!Jestions	4	
Multiplying Polynomials	3	
Recurrences	14	
Chip testing	9	
Total:	34	

SUBMISSION DETAILS

Things to submit:

- Submit your assignment in your Github repo.
  - The written parts of this assignment as a .pdf named "CS5006_[lastname] _A2.pdf". For example, my file would be named "CS5006_Slaughter_A2pdf". (There should be no brackets around your name).
  - Make sure your name is in the document as well (e.g., written on the top of the first page).
  - Make sure your assignment is in the A2 folder in your Github repo.

HELPFUL HINTS

- Ask clarification questions on Piazza.
- Remember, your write-up should convince graders and instructors that you are providing your own work and should showcase your understanding.
- Use the resources page on the course website for supplemental materials.
- In general, problems will be graded both on whether you are taking the right approach and whether you got the right answer. So, show your work and explain your thinking.